POSITION DEPENDENT PREDICTION COMBINATION FOR INTRA-FRAME VIDEO CODING

Amir Said *Xin Zhao* *Marta Karczewicz* *Jianle Chen* *Feng Zou*

Qualcomm Technologies, Inc., San Diego, CA, USA

arXiv:2505.23672v1 [eess.IV] 29 May 2025

ABSTRACT

Intra-frame prediction in the High Efficiency Video Coding (HEVC) standard can be empirically improved by applying sets of recursive two-dimensional filters to the predicted values. However, this approach does not allow (or complicates significantly) the parallel computation of pixel predictions. In this work we analyze why the recursive filters are effective, and use the results to derive sets of non-recursive predictors that have superior performance. We present an extension to HEVC intra prediction that combines values predicted using non-filtered and filtered (smoothed) reference samples, depending on the prediction mode, and block size. Simulations using the HEVC common test conditions show that a 2.0% bit rate average reduction can be achieved compared to HEVC, for All Intra (AI) configurations.

Index Terms— HEVC, video coding, linear prediction, intra-frame coding

1. INTRODUCTION

Intra-frame coding in the HEVC video coding standard employs the conventional combination of predictive and transform coding, where prediction is based on single reference pixel lines, from previously reconstructed blocks [1, 2]. To attain improved compression efficiency while maintaining low decoder complexity, only very simple linear predictors are used, but the encoder can choose among two non-directional and 33 directional predictors.

This approach has proved to be quite effective, but one of its limitations is that the same form of prediction is used for blocks sizes that can range from 4×4 to 64×64, and while exact directional prediction can work well in the small blocks, it is much less effective in larger blocks.

In HEVC this problem is partially addressed by applying a smoothing filter to the reference lines, with a table-based dependence on predictor type and block size [3]. However, this approach does not address the issue that, in reality, less smoothing is needed for pixels near the top-left block borders, and more for pixels further away from the borders. Furthermore, even for blocks of the same size, there is much statistical variability, and additional coding gains can be obtained by allowing the encoder to test more filter choices.

One interesting solution is to apply sets of recursive filters to smooth the predicted values [4, 5, 6, 7], since this can naturally provide, up to a certain degree, the desired smoothing control, and experimental results demonstrate that it can improve compression efficiency significantly.

The computational complexity of the recursive filters can appear very small, if we measure it with conventional techniques. However, in custom hardware and general purpose processors, video applications are now employing parallel computation. Thus, it is advantageous to use techniques that allow a large number of block pixels to be predicted in parallel, simultaneously. Under this consideration, the recursive filtering approach becomes much less attractive.

We propose an extension to HEVC intra prediction, which was accepted to the latest HM-KTA software,[1] called Position-Dependent Prediction Combination (PDPC). It is meant to address all those issues—while maintaining low complexity and efficient parallel implementation—by employing a simple weighted combination of unfiltered and filtered (smoothed) references, where the weights depend on pixel position, prediction mode, and block size.

In the next sections we show how statistical analysis, plus visualization, can help identify what is the type and degree of smoothing needed in the different cases.

2. EXPLOITING GEOMETRIC AND STATISTICAL FEATURES

Fig. 1 shows the HEVC configuration for intra-frame prediction, for 4×4 blocks, and the notation we use. In an $N \times N$ block, we have the pixels values represented by $v[x, y]$, and their set of predicted values by $p_m[x, y]$ with $x, y \in \{0, 1, \ldots, N-1\}$, and $m \in \{0, 1, \ldots, 34\}$ is the index identifying the predictor type (planar, DC, directions 0 to 33). The reference pixels, which are used for prediction, are represented by a two-dimensional array $r[x, y]$, but with the assumption that only values $r[x, -1]$ and $r[-1, y]$, with $x, y \in \{-1, 0, \ldots, 2N-1\}$ are used [3].

The predictor formulas used by HEVC are designed to exploit geometric features in the blocks. Theoretically, we

[1] https://vceg.hhi.fraunhofer.de/svn/svn_HMKTASoftware/tags/ HM-14.0-KTA-2.0/

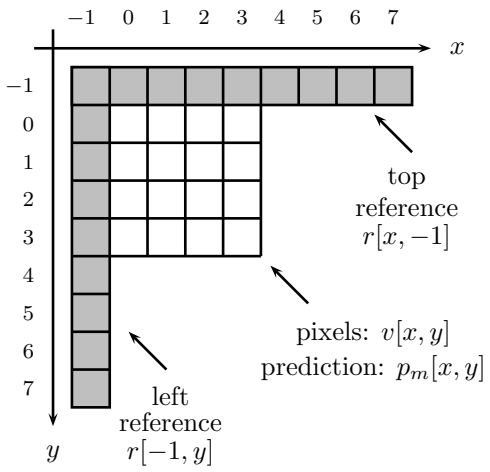

Fig. 1. Configuration used for intra-frame prediction in the HEVC video coding standard.

can improve performance using statistics to design better predictors, but by itself this approach tends to produce solutions that are either not effective, or too complex for the demanding video applications.

We show next that a hybrid approach, exploiting statistical distributions conditioned to geometrical features, can be quite powerful, and even if it cannot be directly used, it yields valuable information on how to design better low complexity predictors.

We can employ the HEVC encoder as a classifier of geometric features, and analyze the statistics of the blocks where a predictor with index m was chosen. To simplify the notation, we represent the block pixel, predicted, and reference values as columns vectors $\mathbf{v}$, $\mathbf{p}_m$, and $\mathbf{r}$, with dimensions N^2, N^2, and $4N+1$, respectively. Using this notation, we can use training to estimate the matrices

$$\mathbf{P}_m = E_{\mathbf{r}|m}\{\mathbf{r}\,\mathbf{r}^\mathsf{T}\}, \quad \mathbf{Q}_m = E_{\mathbf{v},\mathbf{r}|m}\{\mathbf{v}\,\mathbf{r}^\mathsf{T}\}, \tag{1}$$

where operators $E_{\mathbf{r}|m}$ and $E_{\mathbf{v},\mathbf{r}|m}$ define expectation on $\mathbf{v}$ and $\mathbf{r}$, conditioned to m being chosen, by the HEVC encoder, as best predictor index for those block pixel and reference values.

With those statistics we can compute the linear least-squared-error predictor as

$$\mathbf{p}_m = \mathbf{H}_m\mathbf{r}, \quad \mathbf{H}_m = \mathbf{Q}_m\mathbf{P}_m^{-1}. \tag{2}$$

Using matrix $\mathbf{H}_m$ for practical prediction is not feasible because it has $N^2(4N+1)$ elements, and we need one matrix for each predictor index m, and for each block size. However, unlike matrix $\mathbf{Q}_m$, which provides only the level of similarity between pixels and reference values, matrix $\mathbf{H}_m$ can show which similarities are the best to exploit for prediction, by identifying the elements with largest magnitudes in the matrix.

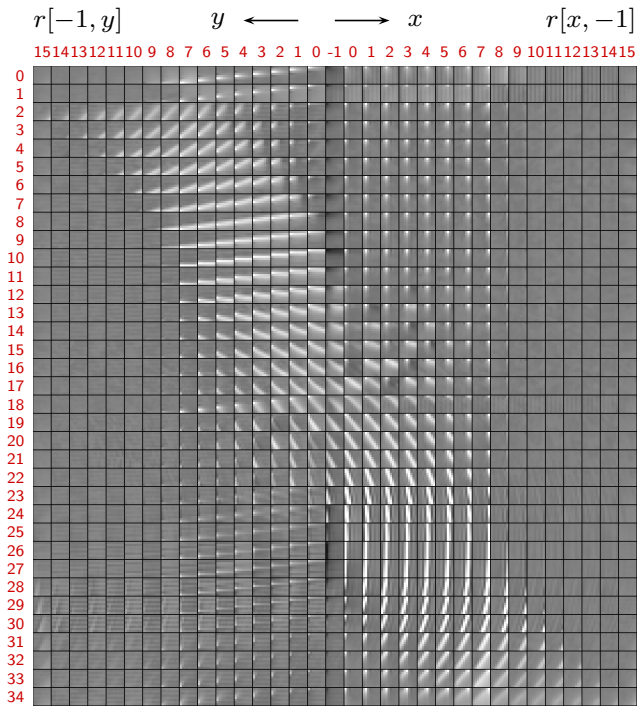

Fig. 2. Example of an image for visualizing the distribution of coefficients in predictor matrices $\mathbf{H}_m$. Zeros are represented as gray, positive as lighter, and negative as darker pixels.

This can be observed by using a simple technique to visualize the properties of the optimal predictors in matrices $\mathbf{H}_m$, and how they vary with m. Fig. 2 shows an example of such $\mathbf{H}_m$ visualization image, where gray level represents the weight of a reference pixel (block columns), for a given predictor with index m (block rows). Each 8×8 block represents a column of $\mathbf{H}_m$, but reorganized to represent the corresponding pixel position.

If we create a similar image, with a predictor matrix $\mathbf{H}$ equal to the HEVC predictors, we would see that for $m = 2,\ldots,34$, we would have well-defined straight lines, according to the direction of those predictors. In Fig. 2 we can observe the same directional patterns (it was, after all, obtained using HEVC classification for training), but we can also observe that the lines have different degrees of "blurring," to compensate for the fact that in real video frames, patterns commonly deviate from perfectly straight lines.

From this observation, the advantage of applying recursive filters to the predicted values become quite clear—it makes the result of HEVC predictions more similar to the optimal (but highly complex) linear predictors in matrices $\mathbf{H}_m$.

Going further, we can observe that most of the coefficients in $\mathbf{H}_m$ are very small, and the important coefficients occur with predictable patterns. In the next section we exploit this fact to design new practical predictors.

3. POSITION-DEPENDENT PREDICTION COMBINATION (PDPC)

There are many ways to approximate the features of the predictors in matrices $\mathbf{H}_m$. In our approach we considered only the solutions that

- Combine statistical analysis with the geometry-based predictors in HEVC to approximate the features of optimal predictors in matrices $\mathbf{H}_m$;
- Avoid radical changes in the prediction methodology, and if possible employ the methods already develped for HEVC;
- Have a formulation that allows parallel computation of all predicted pixels, for all block sizes.

Due to the last requirement, we cannot use recursive filters to vary the support of reference pixels that are used to compute each pixel prediction. Nevertheless, we can obtain similar results if we use more than one set of filtered references, and combine them according to the position of the predicted pixel in a block.

This technique, which we call Position-Dependent Prediction Combination (PDPC), can be used for any number of combinations, but we consider having only the original $\mathbf{r}$, and one additional filtered version of $\mathbf{r}$, which we represent by $\mathbf{s}$. Given any two set of pixel predictors $p_{\mathbf{r}}[x,y]$ and $q_{\mathbf{s}}[x,y]$, computed using only the unfiltered and filtered references $\mathbf{r}$ and $\mathbf{s}$, respectively, the combined predicted value of a pixel, denoted by $p[x,y]$, is defined by (we omit the dependence on m to simplify notation)

$$p[x,y] = c[x,y]p_{\mathbf{r}}[x,y] + (1 - c[x,y])q_{\mathbf{s}}[x,y], \qquad (3)$$

where $c[x,y]$ is a set of weights, computed to approximate predictors in a matrix $\mathbf{H}$.

This general approach can be easily adapted for practical requirements. For example, we can use the predictors already defined by HEVC, denoted by $p_{\mathbf{x}}^{(\text{HEVC})}$, to replace $p_{\mathbf{r}}[x,y]$ and $q_{\mathbf{s}}[x,y]$, and we can define weights using simple formulas.

One additional prediction feature, clearly seen in predictor visualization images as in Fig. 2, is that, even for the directional modes, there is a strong correlation between a pixel value and its nearest pixel in the reference set, which is seen to decay in more or less exponential manner.

Putting these two properties together, one choice that produces a good approximations of features in trained predictors $\mathbf{H}$, reuses HEVC methods, and is easy to compute, is

$$\begin{aligned} p[x,y] &= \frac{c_1^{(v)} r[x,-1] - c_2^{(v)} r[-1,-1]}{2^{y/d_v}} + \\ &+ \frac{c_1^{(h)} r[-1,y] - c_2^{(h)} r[-1,-1]}{2^{x/d_h}} + \\ &+ t[x,y]\; p_{\mathbf{r}}^{(\text{HEVC})} + b[x,y]\; p_{\mathbf{s}}^{(\text{HEVC})} \end{aligned} \qquad (4)$$

where

$$b[x,y] = 1 - \frac{c_1^{(v)} - c_2^{(v)}}{2^{y/d_v}} - \frac{c_1^{(h)} - c_2^{(h)}}{2^{x/d_h}} - t[x,y]. \qquad (5)$$

is a normalization factor, and

$$t[x,y] = \frac{N - \min(x,y)}{N}. \qquad (6)$$

The first terms in eq. (4) are meant to exploit correlations at the block boundaries, similarly to recursive filters, but the fast exponential decay of its weights allow for more flexibility in its combination with other predictor term.

The parameters $c_1^{(v)}$, $c_2^{(v)}$, $c_1^{(h)}$, $c_2^{(h)}$, d_h and d_v can be determined via training (more details shortly). To reduce the number of parameters we also used the rule

$$d_v = d_h = \begin{cases} 1, & \text{if } 16 \times 16 \text{ or smaller} \\ 2, & \text{if } 32 \times 32 \text{ or larger} \end{cases} \qquad (7)$$

Note that when we employ only linear predictors (like HEVC's), we can avoid having to compute both $p_{\mathbf{r}}^{(\text{HEVC})}$ and $p_{\mathbf{s}}^{(\text{HEVC})}$, and use

$$\begin{aligned} p[x,y] &= \frac{c_1^{(v)} r[x,-1] - c_2^{(v)} r[-1,-1]}{2^{y/d_v}} + \\ &+ \frac{c_1^{(h)} r[-1,y] - c_2^{(h)} r[-1,-1]}{2^{x/d_h}} + \\ &+ b'[x,y] p_{\mathbf{s}}^{(\text{HEVC})} \end{aligned} \qquad (8)$$

where

$$b'[x,y] = 1 - \frac{c_1^{(v)} - c_2^{(v)}}{2^{y/d_v}} - \frac{c_1^{(h)} - c_2^{(h)}}{2^{x/d_h}}. \qquad (9)$$

and

$$\mathbf{s} = a\,\mathbf{r} + (1-a)(\mathbf{h}_k * \mathbf{r}). \qquad (10)$$

where a is another parameter obtained from training and $\mathbf{h}_k$ is a selected filter parameterized by index k. In our experimental tests we used this last formulation with binomial filters [8] for $\mathbf{h}_k$, where k is the order of the filter, also optimized from training.

The optimization corresponds to finding the PDPC parameter vector $\boldsymbol{\alpha}$ that, given the data classification defined by the HEVC encoder, minimizes the residual mean squared error. Using this approach we can exploit the fact that, since PDPC is linear, it has a corresponding prediction matrix $\tilde{\mathbf{H}}_m(\boldsymbol{\alpha})$, and the squared error minimization corresponds to

$$\min_{\boldsymbol{\alpha}} \left[\operatorname{tr}\left\{ \tilde{\mathbf{H}}_m(\boldsymbol{\alpha}) \mathbf{P}_m \tilde{\mathbf{H}}_m^{\intercal}(\boldsymbol{\alpha}) \right\} - 2 \operatorname{tr}\left\{ \tilde{\mathbf{H}}_m(\boldsymbol{\alpha}) \mathbf{Q}_m^{\intercal} \right\} \right], \qquad (11)$$

where $\mathbf{P}_m$ and $\mathbf{Q}_m$ are defined in eq. (1).

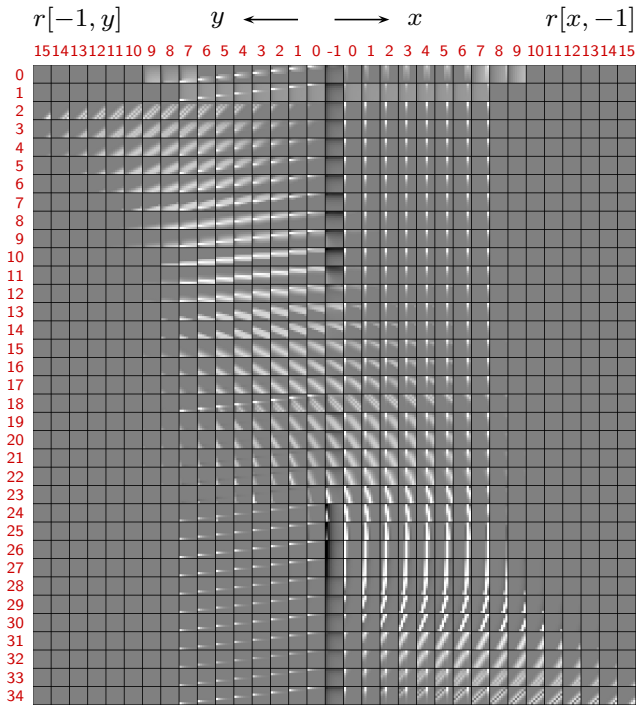

Fig. 3. Image for visualizing the PDPC predictor matrices ($\tilde{\mathbf{H}}_m(\boldsymbol{\alpha})$), where we can observe how they approximate the matrices in Fig. 2, but with a much smaller number of parameters.

Table 1. PDPC coding gains over HEVC, version with four sets of predictor parameters.

	Y	U	V
Class A	–2.24%	–1.80%	–1.83%
Class B	–2.26%	–1.74%	–1.90%
Class C	–1.59%	–1.45%	–1.66%
Class D	–1.54%	–1.29%	–1.38%
Class E	–2.48%	–3.65%	–3.44%
Average	**–2.01%**	**–1.89%**	**–1.97%**

Table 2. PDPC coding gains over HEVC, version with four sets of predictor parameters, UHD 3840×2160 sequences.

UHD Sequence	Y	U	V
EBU LupoBoa	–4.56%	–7.68%	–7.89%
EBU StudioDancer	–4.18%	–6.32%	–6.00%
EBU VeggieFruits	–3.45%	–5.75%	–5.07%
Average	**–4.06%**	**–6.58%**	**–6.32%**

Table 3. PDPC coding gains over HEVC, version with two sets of predictor parameters.

	Y	U	V
Class A	–1.54%	–1.20%	–1.28%
Class B	–1.53%	–1.25%	–1.28%
Class C	–1.29%	–1.00%	–1.05%
Class D	–1.10%	–0.89%	–0.88%
Class E	–1.85%	–2.69%	–2.54%
Average	**–1.45%**	**–1.34%**	**–1.34%**

4. EXPERIMENTAL RESULTS

Using the training procedure described above, we obtained sets of PDPC parameters that approximate the optimal prediction matrices $\mathbf{H}_m$. This can be observed by applying the same visualization technique of Fig. 2 to the PDPC prediction matrices $\tilde{\mathbf{H}}_m(\boldsymbol{\alpha})$, an shown in the example of Fig. 3.

The proposed prediction method has been integrated into the latest HEVC reference software (HM 16.6),[2] and tested using All Intra (AI) coding configuration and main 10 profile, as specified in the the common tests conditions defined during development HEVC, and evaluated using the Bjøntegaard-Delta (BD) coding gain measure [9, 10]. We tested two versions, which employ two and four sets of predictor parameters, tested by the encoder, and the selected optimal one is signaled at Coding Unit (CU) level. In both cases, one of the options is to use the HEVC predictors directly (i.e., with PDPC). A slightly more optimized version is available for testing in the HM-KTA software.

Table 1 shows the results for the version with four sets of predictor parameters. We can observe that PDPC yields an average coding gain around 2%. Even larger gains (around 4%) can be obtained on UHD sequences, as shown in Table 2. Table 3 shows that the version with two sets of parameters yield smaller, but still significant, gains.

5. REFERENCES

[1] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur, "Intra coding of the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1792–1801, Dec. 2012.

[3] M. Wien, *High Efficiency Video Coding: Coding Tools and Specification*, Springer-Verlag, Berlin, 2015.

[2] https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware /tags/HM-16.6/

[4] K. McCann, W.-J. Han, I.-K. Kim, J. Min, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y. M. Hong, M.-S. Cheon, and N. Shlyakhov, "Video coding technology proposal by Samsung (and BBC)," Tech. Rep. JCTVC-A124, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Dresden, Germany, Apr. 2010.

[5] W.-J. Han, J. Min, I.-K. Kim, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y. M. Hong, M.-S. Cheon, N. Shlyakhov, K. McCann, T. Davies, and J.-H. Park, "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1709–1720, Dec. 2010.

[6] Y. Chen, J. Han, and K. Rose, "A recursive extrapolation approach to intra prediction in video coding," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, Vancouver, BC, Canada, May 2013, pp. 1734–1738.

[7] S. Li, Y. Chen, J. Han, T. Nanjundaswamy, and K. Rose, "Rate-distortion optimization and adaptation of intra prediction filter parameters," in *Proc. IEEE Int. Conf. Image Process.*, Paris, France, Oct. 2014, pp. 3146–3150.

[8] M. Aubury and W. Luk, "Binomial filters," *J. VLSI Signal Process.*, vol. 12, no. 1, pp. 1–8, 1995.

[9] F. Bossen, "Common test conditions and software reference configurations," Tech. Rep. JCTVC-H1100, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, San Jose, CA, Feb. 2011.

[10] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," Tech. Rep. VCEG-M33, ITU-T SG16 Q.6, Austin, TX, Apr. 2001.